C1

2. Patent Application No. 09/751,653, entitled "A Virtual Path Restoration Scheme Using Fast Dynamic Mesh Restoration In An Optical Network" having Ali N. Saleh, H. Michael Zadikian, Zareh Baghdasarian, and Vahid Parsi as inventors, filed December 30, 2000.

3. Patent Application No. 09/858,743, entitled "Resource Reservation Scheme For Path Restoration In An Optical Network" having Ali N. Saleh, H. Michael Zadikian, Zareh Baghdasarian, and Vahid Parsi as inventors, filed May 16, 2001.

4. Patent Application No. 09/859,166, entitled "Method For Restoring A Virtual Path In An Optical Network Using 1+1 Protection" having Ali N. Saleh, H. Michael Zadikian, Zareh Baghdasarian, and Vahid Parsi as inventors, filed May 16, 2001.

These applications are assigned to Cisco Technology, Inc., the assignee of the present invention, and are hereby incorporated by reference, in their entirety and for all purposes.

Please replace the paragraph on page 5, lines 6-15 with the following paragraph:

C2

An OXC can be either transparent (purely optical, in which the signals are never converted from optical signals) or opaque (in which the optical signals are converted from optical signals into electrical signals, switched, and then converted back into optical signals). Transparent optical cross connects provide little in the way manageability because the information is never made accessible to the OXC's operator. In contrast, opaque OXCs can be configured to permit access to the information being switched. However, neither type of OXC maintains information regarding the topology of the network and, in fact, OXCs possess no intrinsic network intelligence. Moreover, OXC technology is expensive, making initial investment quite high, as well as the cost of future expansion.

Please replace the paragraph on page 7, lines 21-26 with the following paragraph:

C3

A routing protocol that supports relatively simple provisioning and relatively fast restoration (on the order of , for example, 50 ms), while providing relatively efficient bandwidth usage (i.e., minimizing excess bandwidth requirements for restoration, on the order of less than 100% redundant capacity and preferably less than 50% redundant capacity). Such a routing protocol is, in one embodiment, easily be scaled to accommodate increasing bandwidth requirements.

Please replace the paragraph beginning on page 7, line 28, and ending on page 8, line 7 with the following paragraph:

C4

According to one embodiment of the present invention, an apparatus and method are described for configuring routes over a network. Such a method, embodied in a protocol of the present invention, provides several advantages. A protocol according to the present invention provides relatively fast restoration (on the order of 50 ms), while providing relatively efficient bandwidth usage (i.e., minimizing excess bandwidth requirements for restoration on the order of less than 100% redundant capacity and preferably, less than 50% redundant capacity). Moreover, a protocol according to one embodiment of the present invention scales well to accommodate increasing bandwidth demands of the services being supported.

Please replace the paragraph on page 8, lines 8-16 with the following paragraph:

In one embodiment of the present invention, a method of operating an optical network is described. The network includes a number of nodes coupled by a number of links. A method according to this embodiment of the present invention provisions a virtual path between a first and a second one of the plurality of nodes by: identifying the first and the second nodes, discovering a physical path from the first node to the second node, and establishing the virtual path. The method discovers a physical path from the first node to the second node by automatically identifying nodes forming the physical path. The method

establishes the virtual path by configuring a set of connections between the nodes forming the physical path.

C4

Please replace the paragraph on page 8, lines 17-22 with the following paragraph:

In another embodiment of the present invention, a method is described that terminates the virtual path by sending a termination message from one of the first and second nodes to the other of the first and second nodes. The termination message is sent along the physical path and resources for the virtual path are deallocated by each one of the nodes forming the physical path as the termination message is sent to the next one of the nodes that form the physical path.

Please replace the paragraph beginning on page 8, line 23 and ending on page 9, line 4 with the following paragraph:

In yet another embodiment of the present invention, a method is described that restores a virtual path in response to a failure along the physical path created between a first node and a second node by a provisioning operation such as that described above (although a virtual path restored by a method according to the present invention may be provisioned in any manner deemed desirable). Such a method begins by discovering an alternate physical path from the first node to the second node. The alternate physical path is discovered by automatically identifying nodes forming the alternate physical path. This may be based on any number of criteria, such as cost, quality of service, latency, or other metric. The method then re-establishes the virtual path by configuring a set of connections between the nodes forming the alternate physical path. This may require an entirely new end-to-end alternate physical path, or may simply be the addition of a node or link to the existing physical path.

Please replace the paragraph on page 11, lines 14-25 with the following paragraph:

C5

In one embodiment, a routing protocol is described that provides many advantages, including restoration times on the order of 50 ms or less (e.g., comparable to those of SHRs)

and relatively high utilization efficiency (e.g. by reducing the amount redundant bandwidth, preferably to, for example, 50% or less). The protocol achieves the former by using a physical network layer (e.g., SONET) for communications between network nodes. Preferably, no other protocols are interspersed between the routing protocol and the transmission medium. Also preferably, all protocol-related status and control messages are communicated in-band (e.g., carried by the physical network layer, for example, in certain of a SONET frame's overhead bytes), which allows events to be sent between network nodes at hardware speeds. However, out-of-band communication channels can also be successfully employed to carry such information.

Please replace the paragraph beginning on page 11, line 26, ending on page 12, line 11 with the following paragraph:

Another mechanism employed by the protocol to improve restoration time is distributed intelligence, which also supports end-to-end provisioning. The protocol, in one embodiment, relies on a distributed routing protocol, which employs event pipelining and parallel execution of protocol processes. Because multiple actions occur in parallel, event delays are minimized. In such an embodiment, the protocol also uses a distributed database and relies on distributed control to restore failures. In one embodiment, every node maintains an up-to-date view of network topology, (i.e., available nodes and links, and configured connections). Changes that occur in the network, whether caused by failed links, newly provisioned connections, or added/failed/removed nodes, are "broadcast" throughout the network, using special protocol packets and procedures. Topology distribution normally runs concurrently with, and in parallel to, failure restoration activities, but at a much lower priority.

Please replace the paragraph on page 12, lines 13-20 with the following paragraph:

This is achieved by making the protection bandwidth a user-configurable parameter, and attaching a priority (or QoS) metric to all configured connections (referred to herein as virtual paths or VPs) and links. The QoS parameter makes it possible to reduce the required percentage of protection bandwidth even further, while maintaining the same quality of service for those connections that need and, more importantly, can afford such treatment.

C5 cancel

Thus, availability is mapped into a cost metric and only made available to users who can justify the cost of a given level of service.

Please replace the paragraph on page 13, lines 1-11 with the following paragraph:

C6

Nodes that attach to multiple zones are referred to herein as border nodes. Border nodes are required to maintain a separate topological database, also called a link-state or connectivity database, for each of the zones they attach to. Border nodes use the connectivity database(s) for intra-zone routing. Border nodes are also required to maintain a separate database that describes the connectivity of the zones themselves. This database, which is called the network database, is used for inter-zone routing. It describes the topology of a special zone, referred to herein as the backbone, which is always assigned an ID of 0. The backbone has all the characteristics of a zone. There is no need for a backbone's topology to be known outside the backbone, and its border nodes need not be aware of the topologies of other zones.

Please replace the paragraph on page 14, lines 4-14 with the following paragraph:

C7

As noted, the protocol routes information at two different levels: inter-zone and intra-zone. The former is only used when the source and destination nodes of a virtual path are located in different zones. Inter-zone routing supports path restoration on an end-to-end basis from the source of the virtual path to the destination by isolating failures between zones. In the latter case, the border nodes in each transit zone originate and terminate the path-restoration request on behalf of the virtual path's source and destination nodes. A border node that assumes the role of a source (or destination) node during the path restoration activity is referred to herein as a proxy source (destination) node. Such nodes are responsible for originating (terminating) the RPR request with their own zones. Proxy nodes are also required to communicate with border nodes in other zones to establish an inter-zone path for the VP.

Please replace the paragraph beginning on page 14, line 24, ending on page 15, line 2 with the following paragraph:

C8

Fig. 1 illustrates the layout of a node ID 100 using three types of node IDs. As shown in Fig. 1, a field referred to herein as type ID 110 is allocated either one or two bits, a zone ID 120 of between 2-6 bits in length, and a node address 130 of between about 8-13 bits in length. Type 0 IDs allocate 2 bits to zone ID and 13 bits to node address, which allows up to 2^{13} or 8192 nodes per zone. As shown in Fig. 1, type 1 IDs devote 4 bits to zone ID and 10 bits to node address, which allows up to 2^{10} (i.e. 1024) nodes to be placed in each zone. Finally, type 2 IDs use a 6-bit zone ID and an 8-bit node address, as shown in Fig. 1. This allows up to 256 nodes to be addressed within the zone. It will be obvious to one skilled in the art that the node ID bits can be apportioned in several other ways to provide more levels of addressing.

Please replace the paragraph on page 18, lines 24-30 with the following paragraph:

C9

Once adjacency between two neighbors has been established, the nodes periodically exchange Hello packets. The interval between these transmissions is a configurable parameter that can be different for each link, and for each direction. Nodes are expected to use the *HelloInterval* parameters specified in their neighbor's Hello message. A neighbor is considered dead if no Hello message is received from the neighbor within the *HelloDeadInterval* period (also a configurable parameter that can be link -and direction-specific).

Please replace the paragraph on page 19, lines 15-27 with the following paragraph:

C10

During normal network operation, the originating node of an LSA transmits LS update messages when the node detects activity that results in a change in its LSA. The node sets the HOP_COUNT field of the LSA to 0 and the LSID field to the LSID of the previous instance plus 1. Wraparound may be avoided by using a sufficiently-large LSID (e.g., 32 bits). When another node receives the update message, the node records the LSA in its database and

C10

schedules it for transmission to its own neighbors. The HOP_COUNT field is incremented by one and transmitted to the neighboring nodes. Likewise, when the nodes downstream of the current node receive an update message with a HOP_COUNT of H, they transmit their own update message to all of their neighbors with a HOP_COUNT of H+1, which represents the distance (in hops) to the originating node. This continues until the update message either reaches a node that has a newer instance of the LSA in its database or the hop-count field reaches MAX_HOPS.

Please replace the paragraph on page 24, lines 4-17 with the following paragraph:

C11

Otherwise, the node's link state database is searched to find the current LSA (step 640), and if not found, the current LSA is written into the database (step 645). If the current LSA is found in the link state database, the current LSA and the LSA in the database are compared to determine if they were sent from the same node (step 650). If the LSAs were from the same node, the LSA is installed in the database (step 655). If the LSAs were not from the same node, the current LSA is compared to the existing LSA to determine which of the two is more recent (step 660). The process for determining which of the two LSAs is more recent is discussed in detail below in reference to Fig. 7. If the LSA stored in the database is the more recent of the two, the LSA received is simply discarded (step 665). If the LSA in the database is less recent than the received LSA, the new LSA is installed in the database, overwriting the existing LSA (step 670). Regardless of the outcome of this analysis, the LSA is then acknowledged by sending back an appropriate response to the node having transmitted the Hello message (step 675).

Please replace the paragraph on page 24, lines 18-28 with the following paragraph:

Fig. 7 illustrates one method of determining which of two LSAs is the more recent. An LSA is identified by the Node ID of its originating node. For two instances of the same LSA, the process of determining the more recent of the two begins at step 700 by comparing the LSAs LSIDs. In one embodiment of the protocol, the special ID *FIRST_LSID* is considered to be higher than any other ID. If the LSAs LSIDs are different, the LSA with the

higher LSID is the more recent of the two (step 710). If the LSAs have the same LSIDs, then HOP_COUNTs are compared (step 720). If the HOP_COUNTs of the two LSAs are equal then the LSAs are identical and neither is more recent than the other (step 730). If the HOP_COUNTs are not equal, the LSA with the lower HOP_COUNT is used (step 740). Normally, however, the LSAs will have different LSIDs.

C11

Please replace the paragraph on beginning on page 24, line 29 and ending on page 25, line with the following paragraph:

The basic flooding mechanism in which each packet is sent to all active neighbors except the one from which the packet was received can result in an exponential number of copies of each packet. This is referred to herein as a broadcast storm. The severity of broadcast storms can be limited by one or more of the following optimizations:

Please replace the paragraph on page 25, lines 10-14 with the following paragraph:

C12

3. Nodes can be prohibited from generating more than one new instance of an LSA every *MinLSAInterval* interval (a minimum period defined in the LSA that can be used to limit broadcast storms by limiting how often an LSA may be generated or accepted (See Fig. 15 and the accompanying discussion)).

Please replace the paragraph on page 34, lines 21-24 with the following paragraph:

C13

1. The origin node builds a shortest path first (SPF) tree with "self" as root. Prior to building the SPF tree, the link-state database is pruned of all links that either don't have enough (available) bandwidth to satisfy the request, or have been assigned a QoS level that exceeds that of the VP being restored.

Please replace the table on page 36, lines 11-35 with the following table:

Field	Usage
Origin Node	The Node ID of the node that originated this request. This is either the source node of the VP or a proxy border node.
Target Node	Node ID of the target node of the restore path request. This is either the destination node of the VP or a proxy border node.
Received From	The neighbor from which we received this message.
First Sequence Number	Sequence number of the first received copy of the corresponding restore-path request.
Last Sequence Number	Sequence number of the last received copy of the corresponding restore-path request.
Bandwidth	Requested bandwidth
QoS	Requested QoS
Timer	Used by the node to timeout the RPR
T-Bit	Set to 1 when a Terminate indicator is received from any of the neighbors.
Pending Replies	Number of the neighbors that haven't acknowledged this message yet.
Sent To	A list of all neighbors that received a copy of this message. Each entry contains the following information about the neighbor: *AckReceived*: Indicates if a response has been received from this neighbor. *F-Bit*: Set to 1 when *Flush* indicator from this neighbor.

Table 6. RPR Fields

C 14

Please replace the table on page 40, with the following table:

Response Type	*Flush Indicator?*	*Terminate Indicator?*	*Received Sequence Number*	*Action*
X	X	X	Not Valid	Ignore response
Negative	No	No	is not equal to Last	Ignore response
Negative	X	No	is equal to Last	Release bandwidth allocated for the VP on the link the response was received on
Negative	Yes	No	Valid	Release bandwidth allocated for the VP on the link that the response was received on
Negative	X	Yes	Valid	Release all bandwidth allocated for the VP

C 15

Positive	X	X	Valid	Commit bandwidth allocated for the VP on the link the response was received on; release all other bandwidth.

Table 7. Actions taken by a tandem node upon receiving an RPR.

Please replace the paragraph on page 42, lines 8-21 with the following paragraph:

If a *Terminate* was specified in the RPR response (step 1240), the bandwidth on all links over which the RPR was forwarded is freed (step 1245) and the *Terminate* and *Flush* bits from the RPR response are saved in the RPRE (step 1250). If a *Terminate* was not specified in the RPR response, bandwidth is freed only on the input link (i.e., the link from which the response was received) (step 1255), the *Terminate* and *Flush* bits are saved in the RPRE (step 1260), and the *Flush* bit of the RPR is cleared (step 1265). If a *Terminate* was not specified in the RPR, the *Pending Replies* field in the RPRE is decremented, (step 1270). If this field remains non-zero after being decremented the process completes. If *Pending Replies* is equal to zero at this point, or a *Terminate* was not specified in the RPR, the RPR is sent to the node specified in the RPR's *Received From* field (i.e. the node that sent the corresponding request) (step 1280). Next, the bandwidth allocated on the link to the node specified in the RPR's *Received From* field is released (step 1285) and an RPR deletion timer is started (step 1290).

Please replace the paragraph beginning on page 42, line 22, ending on page 43, line 9 with the following paragraph:

Fig. 13 illustrates the steps taken in processing positive RPR responses. The processing of positive RPR responses begins at step 1300 with a search of the local database to determine whether an RPRE corresponding to the RPR response is stored therein. If a corresponding RPRE cannot be found, the RPR response is ignored (step 1310). If the RPR response RPRE is found in the local database, the input link is verified as being consistent with the path stored in the RPR (step 1320). If the input link is not consistent with the RPR path, the RPR response is ignored once again (step 1310). If the input link is consistent with

C16

path information in the RPR, the next hop information specified in the RPR response path is compared with the *Received From* field of the RPRE (e.g., *Response.Path[Response.PathIndex + 1] != RPRE.ReceivedFrom*) (step 1330). If the next hop information is not consistent, the RPR response is again ignored (step 1310). However, if the RPR response's next hop information is consistent, bandwidth allocated on input and output links related to the RPR is committed (step 1340). Conversely, bandwidth allocated on all other input and output links for that VP is freed at this time (step 1350). Additionally, a positive response is sent to the node from which the RPR was received (step 1360), and an RPR deletion timer is started (step 1370) and the local matrix is configured (step 1380).

Please replace the paragraph on page 44, lines 6-18, with the following paragraph:

C17

Further optimizations of the protocol can easily be envisioned by one of skill in the art, and are intended to be within the scope of this specification. For examples in one embodiment, a mechanism to further reduce the amount of broadcast traffic generated for any given VP. In order to prevent an upstream neighbor from sending the same instance of an RPR every T milliseconds, a tandem node can immediately return a no-commit positive response to that neighbor, which prevents it from sending further copies of the instance. The response simply acknowledges the receipt of the request, and doesn't commit the sender to any of the requested resources. Preferably, however, the sender (of the positive response) periodically transmits the acknowledged request until a valid response is received from its downstream neighbor(s). This mechanism implements a piece-wise, or hop-by-hop, acknowledgment strategy that limits the scope of retransmitted packets to a region that gets progressively smaller as the request gets closer to its target node.

Please replace the paragraph on page 45, lines 3-9, with the following paragraph:

C18

The above strategy is not the preferred method of handling link errors in the present invention. This is because the fast restoration times required dictates that 2-way, end-to-end communication be carried out in less than 50ms. A drawback of the above-described solution is the time wasted while waiting for an acknowledgment to come back from the receiving

node. A safe timeout period for a 2000 mile span, for instance, is over 35ms, which doesn't leave enough time for a retransmission in case of an error.

Please replace the paragraph on page 47, line 5, with the following paragraph:

C19

Table 9A. Configured VPs.

Please replace the paragraph on page 48, line 35, with the following paragraph:

C20

b. Else, copy column **h**-1 to column **h**

Please replace the paragraph on page 49, lines 1-2, with the following paragraph:

C21

c. For each node **n** in *Ready* (do not include nodes added during this iteration of the loop):

Please replace the paragraph on page 50, lines 18-28, with the following paragraph:

C22

Fig. 16 illustrates the layout of a header 1600. Shown therein is a request response indicator (RRI) 1610, a negative response indicator (NRI), a terminate/commit path indicator (TPI) 1630, a flush path indicator (FPI) 1640, a command field 1650, a sequence number (1660), an origin node ID (1670) and a target node ID (1680). A description of these fields is provided below in Table 10. It will be notes that although the terms "origin" and "target" are used in describing header 1600, their counterparts (source and destination, respectively) can be used in their stead. Preferably, packets sent using a protocol according to the present invention employ a header layout such as that shown as header 1600. Header 1600 is then followed by zero or more bytes of command specific data, the format of which, for certain commands, is shown in Figs. 17-21 below.

Please replace the table on page 52 with the following table:

Command Name	Command Code	Description
INIT	0	Initialize Adjacency
HELLO	1	Used to implement the Hello protocol (see Section 3 for more details).
RESTORE_PATH	2	Restore Virtual Path or VP
DELETE_PATH	3	Delete and existing Virtual Path
TEST_PATH	4	Test the specified Virtual Path
LINK_DOWN	5	Used by slave nodes to inform their master(s) of local link failures
CONFIGURE	6	Used by master notes to configure slave nodes.
GET_LSA	7	Get LSA information from other nodes
CREATE_PATH	8	Create Virtual Path

Table 11. Exemplary protocol commands.

Please replace the paragraph on page 52, lines 7-15, with the following paragraph:

Fig. 17 illustrates the layout of command specific data for an initialization packet 1700 which in turn causes a START event to be sent to the Hello State Machine of the receiving node. Initialization packet 1700 includes a node ID field 1710, a link cost field 1720, one or more QoS capacity fields (as exemplified by QoS3 capacity (Q3C) field 1730 and a QoSn capacity (QnC) field 1740), a Hello interval field 1750 and a time-out interval field 1760. It should be noted that although certain fields are described as being included in the command-specific data of initialization packet 1700, more or less information could easily be provided, and the information illustrated in Fig. 17 could be sent using two or more types of packets.

Please replace the paragraph beginning on page 54, line 15, and ending on page 55, line 9 with the following paragraph:

Fig. 19 illustrates the layout of command-specific data for a GET_LSA packet 1900 of a protocol according to the present invention. GET_LSA packet 1900 has its first byte set to zero (exemplified by a zero byte 1905). GET_LSA packet 1900 includes an LSA count 1910 that indicates the number of LSAs being sought and a node ID list 1920 that reflects one or more of the node IDs for which an LSA is being sought. Node ID list 1920 includes node IDs

1930(1)-(N). The GET_LSA response contains a mask that contains a "1" in each position for which the target node possesses an LSA. The low-order bit corresponds to the first node ID specified in the request, while the highest-order bit corresponds to the last possible node ID. The response is then followed by one or more Hello messages that contain the actual LSAs requested.

Please replace the paragraph beginning on page 55, line 22, and ending on page 56, line 3 with the following paragraph:

The Restore Path packet is sent by source nodes (or proxy border nodes), to obtain an end-to-end path for a VP. The packet is usually sent during failure recovery procedures but can also be used for provisioning new VPs. The node sending the RPR is called the origin or source node. The node that terminates the request is called the target or destination node. A restore path instance is uniquely identified by its origin and target nodes, and VP ID. Multiple copies of the same restore-path instance are identified by the unique sequence number assigned to each of them. Only the sequence number need be unique across multiple copies of the same instance of a restore-path packet. Table 17 provides the definitions for the fields shown in Fig. 20.

C26

Please replace the paragraph on page 56, lines 8-14 with the following paragraph:

Fig. 21 illustrates the layout of command-specific data for a CREATE_PATH (CP) packet 2100. CP packet 2100 includes a virtual path identifier (VPID) field 2110, a checksum field 2120, a path length field 2130, a HOP_COUNT field 2140, and an array of path lengths (exemplified by a path field 2150). Path field 2150 may be further subdivided into hop fields (exemplified by hop fields 2160 (1)-(N), where N may assume a value no larger than MAX_HOPS).

C27 cont.

c27 cancel

Please replace the paragraph beginning on page 56, line 15, and ending on page 57, line 2 with the following paragraph:

The CP packet is sent by source nodes (or proxy border nodes), to obtain an end-to-end path for a VP. The node sending the CP is called the origin or source node. The node that terminates the request is called the target or destination node. A CP instance is uniquely identified by its origin and target nodes, and VP ID. Multiple copies of the same CP instance are identified by the unique sequence number assigned to each of them. Only the sequence number need be unique across multiple copies of the same instance of a restore-path packet. Table 18 provides the definitions for the fields shown in Fig. 21.

Please replace the paragraph on page 57, line 5 with the following paragraph:

c28

Table 18. Field definitions for a Create Path packet.

In accordance with 37 CFR § 1.121(c)(1)(iii), Appendix B provides marked up versions of the amended paragraphs illustrating the newly introduced changes in the specification.